TEC | Tecnológico de Costa Rica

Programa del curso IC-2101

Programación Orientada a Objetos

Escuela de Computación
Carrera de Ingeniería en Computación, Plan 410.

TEC | Tecnológico de Costa Rica

I parte: Aspectos relativos al plan de estudios

1 Datos generales

Nombre del curso:	Programación Orientada a Objetos
Código:	IC-2101
Tipo de curso:	Taller
Electivo o no:	No
Nº de créditos:	3
Nº horas de clase por semana:	4
Nº horas extraclase por semana:	5
Ubicación en el plan de estudios:	Curso del II Semestre del Bachillerato de Ingeniería en Computación
Requisitos:	IC-1802 Introducción a la Programación IC-1803 Taller de Programación
Correquisitos:	Ninguno.
El curso es requisito de:	Ninguno
Asistencia:	Obligatoria
Suficiencia:	No
Posibilidad de reconocimiento:	No
Vigencia del programa:	I Semestre de 2024

TEC | Tecnológico de Costa Rica

2 Descripción general

El curso introduce conceptos y técnicas para la creación de programas de software medianos. Los estudiantes aprenden herramientas y técnicas de programación modular y orientada a objetos, basados en el reconocimiento y la descripción de abstracciones (jerarquías, clases, objetos e instancias), haciendo énfasis en la separación de la interfaz y la implementación.

En el curso se estudia el paradigma de la orientación a objetos, que incluye los principios fundamentales de abstracción, encapsulamiento, modularidad y jerarquía, para construir aplicaciones de software de múltiples capas en el lenguaje de programación Java

3 Objetivos

Objetivo General

Entender las actividades de construcción de software vía su aplicación práctica en el desarrollo de programas medianos.

Aprender los conceptos, las herramientas y las técnicas que se deben utilizar para la construcción de software. Se utiliza principalmente el paradigma de la orientación a objetos y la programación modular, haciendo énfasis en el modelaje y la separación entre interfaz e implementación.

Objetivos Específicos

1. Aplicar el paradigma de la orientación a objetos en la implementación de aplicaciones de software.
2. Identificar las clases, relaciones (asociaciones), operaciones y comportamientos claves del dominio del problema.
3. Diseñar una estructura de módulos y clases para solucionar el problema, evaluando alternativas, en busca de favorecer algunas características deseables (lograr modularidad, reducir acoplamiento, aumentar cohesión).
4. Usar patrones básicos en la solución propuesta.
5. Utilizar el patrón de arquitectura de múltiples capas en la construcción de aplicaciones de software.
6. Escribir programas para satisfacer la especificación de una solución, usando buenas prácticas como el uso de estándares, la aplicación de principios de diseño y el uso de buen estilo de codificación.

7. Trabajar en pequeños equipos, colaborando en los aspectos del desarrollo de software mediante el intercambio de ideas constructivo y organizado.
8. Producir documentación técnica de los módulos y clases que sustentan una aplicación de software.
9. Valorar temas de ingeniería en el desarrollo del software, tales como la importancia de centrarse en los intereses del usuario, seguir un proceso sistemático y trabajar con recursos limitados

4 Contenidos

1. Introducción

1.1. ¿Qué es la Ingeniería del software?
1.2. Involucrados en la ingeniería del software
1.3. Principios de ingeniería del software
1.4. Procesos y proyectos de software
1.5. Modelaje en el desarrollo de software

2. Orientación a Objetos

2.1. Introducción a la Orientación a objetos
2.2. Transición desde el paradigma procedimental
2.3. Conceptos del paradigma orientado a objetos
  2.3.1. Abstracción
  2.3.2. Encapsulamiento
  2.3.3. Modularidad
  2.3.4. Jerarquía

3. Orientación a Objetos en Java

3.1. Objeto
3.2. Clase
3.3. Atributo
3.4. Método
3.5. Mensaje
3.6. Modificadores de acceso
3.7. Visibilidad
3.8. Módulos (paquetes)
3.9. Encapsulamiento y principios de ocultamiento
3.10. Vista rápida a las características del lenguaje
3.11. Declaración de clases
  3.11.1. Instanciación, constructor y destructor
  3.11.2. Variables de instancia o atributos
  3.11.3. Variables locales
  3.11.4. Métodos, sobrecarga de métodos y parámetros
  3.11.5. Visibilidad de clases, atributos y métodos
  3.11.6. Métodos de clase y uso de static.

3.11.7. Primitivas del lenguaje
3.12. Administración de la memoria y recolección de basura
3.13. Estándares de codificación (clases, atributos y métodos)
3.14. El Entorno de desarrollo Integrado (IDE)
3.15. Perspectiva de codificación y depuración en el IDE.
3.16. Plataforma Java y sus distintos sabores (J2ME, J2SE, J2EE)

4. Relaciones entre objetos
4.1. Asociación
4.2. Agregación
4.3. Composición
4.4. Dependencia
4.5. Representación de relaciones
4.6. Diagrama de clases (de alto y bajo nivel)
4.7. Representación en Java de los modelos en UML
4.8. Diagramas de UML para representar la estructura estática y dinámica de una aplicación de software
4.9. Determinar la relación entre interesados y los diagramas de estructura estática y dinámica

5. Relaciones entre clases
5.1. Estructura de clases
5.2. Herencia
5.3. Redefinición de métodos
5.4. Clases abstractas
5.5. Herencia y clases abstractas
5.6. Polimorfismo
5.7. Java interface
5.8. Serialización de objetos
5.9. Ligado: estático y dinámico
5.10. Clases y métodos final
5.11. Conversión de tipos (Ensanchamiento y Estrechamiento)

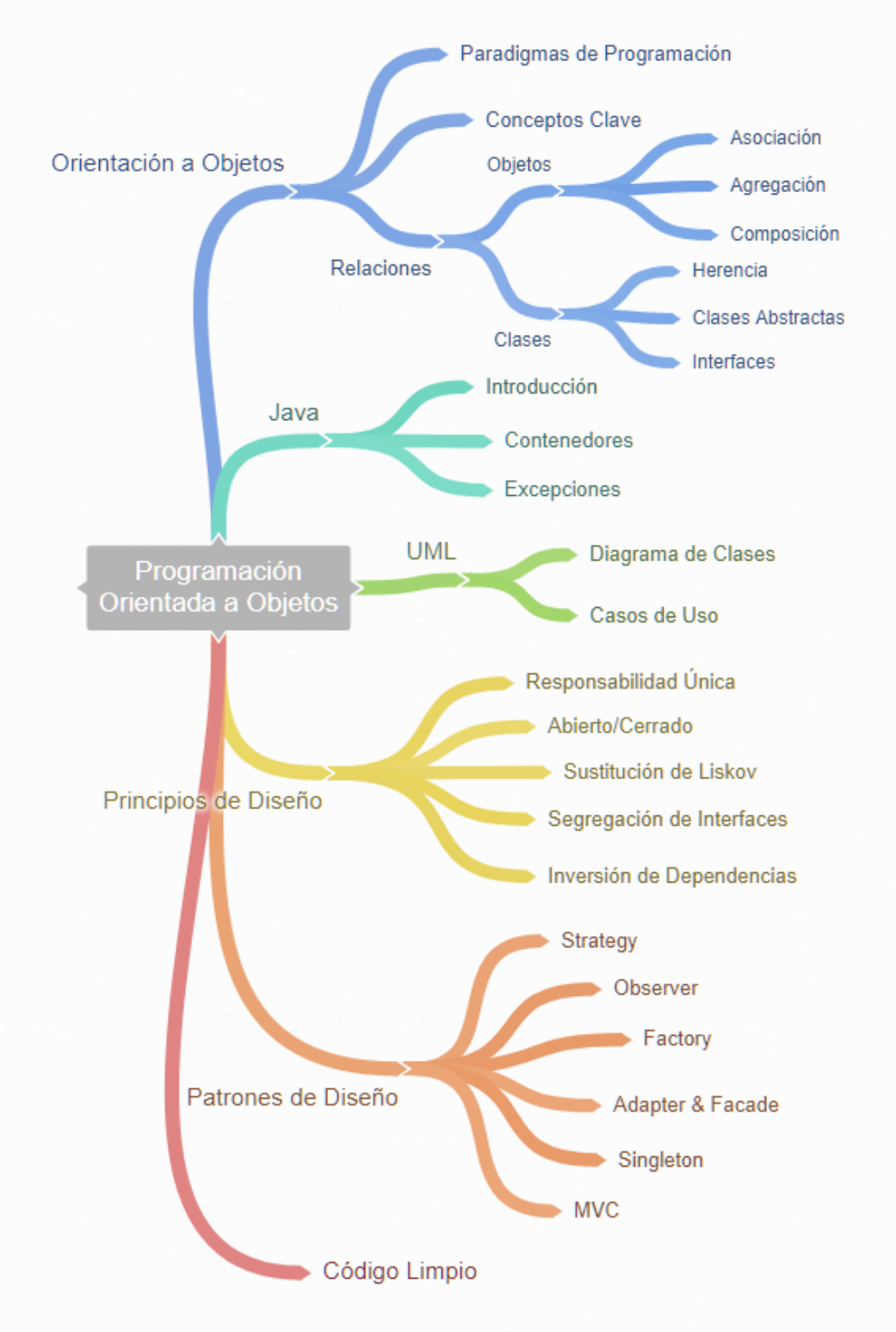
6. Introducción al modelado de software
6.1. Técnicas y heurísticas de modelado
6.2. Principios generales de la descomposición
6.3. Técnicas fundamentales de la descomposición
6.4. Identificación de componentes
6.5. Contratos de operaciones
6.6. Diagramas de interacción (colaboración, secuencia)
6.7. Diagramas de transición (de estados) y de actividad
6.8. Implementación de clases basadas en diagramas de interacción, de transiciones y de actividad
6.9. Del modelo a los programas
6.10. Atributos de calidad ISO/IEC 9126

7. **Excepciones**
   7.1. Concepto y utilidad de las Excepciones
   7.2. Robustez de las aplicaciones de software
   7.3. Jerarquía de Excepciones
   7.4. Tipos de Excepciones: Error, Exception y Runtime Exception
   7.5. Elementos Try/Catch/Finally
   7.6. Implementación de Excepciones de Usuario
   7.7. Manejo de Excepciones

8. **Patrones básicos de diseño**
   8.1. Patrón unitario (singleton)
   8.2. Patrón observador
   8.3. Patrón adaptador (adapter)
   8.4. Patrón fábrica (factory)
   8.5. Patrón Arquitectónico MVC
   8.6. Arquitecturas Multicapas

9. **Introducción al diseño de software**
   9.1. Principios que conducen a un buen diseño
   9.2. Técnicas para tomar buenas decisiones de diseño
   9.3. Aplicación de modelos en el diseño
   9.4. Realización de un buen documento de diseño

10. **¿Qué es la arquitectura del software?**


Programación Orientada a Objetos
Orientación a Objetos
Paradigmas de Programación
Conceptos Clave
Objetos
Asociación
Agregación
Composición
Relaciones
Herencia
Clases Abstractas
Clases
Interfaces
Java
Introducción
Contenedores
Excepciones
UML
Diagrama de Clases
Casos de Uso
Principios de Diseño
Responsabilidad Única
Abierto/Cerrado
Sustitución de Liskov
Segregación de Interfaces
Inversión de Dependencias
Patrones de Diseño
Strategy
Observer
Factory
Adapter & Facade
Singleton
MVC
Código Limpio

II parte: Aspectos operativos

5 Metodología de enseñanza y aprendizaje

La metodología del curso consistirá en la combinación de clases magistrales por parte del profesor con la incorporación de actividades como lecturas, investigaciones y la aplicación de herramientas para la codificación, edición y depuración de programas, y la graficación de diagramas usando la notación UML. Se dejarán tareas cortas programadas que refuercen los temas vistas en la clase.

Se desarrollarán proyectos de programación para aplicar los conceptos y técnicas desarrollados en clase.

Se espera del estudiante una participación activa en general en el curso, haciendo lecturas de las fuentes recomendadas, desarrollando ejercicios y tareas relacionados con lo visto en el curso y participando en el desarrollo de los proyectos de construcción de software.

6 Evaluación

El curso será evaluado mediante tres rubros principales: proyectos, tareas cortas y exámenes cortos.

Proyectos (40%):

Se llevarán a cabo dos proyectos que pueden ser programados o de desarrollo de documentación. Se realizarán de forma grupal y están programados para entregarse en las semanas 9 y 17.

Tareas cortas (40%):

Durante el semestre se estarán realizando un mínimo de ocho tareas cortas. Las tareas pueden requerir que el estudiante realice alguna actividad previa como lecturas o videos. Las tareas pueden abarcar diferentes tipos de trabajos como resúmenes, esquemas, investigaciones, exposiciones, tareas programadas. El 40% del rubro final se distribuye uniformemente entre todas las actividades que se realicen durante el semestre.

Exámenes cortos (20%):

Pruebas cortas que evalúan la comprensión de la teoría vista en clase. Se realizarán un mínimo de diez exámenes cortos, a partir de la semana 2.

Rubro	Porcentaje
Tareas	40%
Exámenes cortos	20%
Proyectos	40%
Total	100%

Cronograma de Actividades

Actividad	Semanas
Introducción	1
Orientación a Objetos	1
Orientación a Objetos en Java	2
Relaciones entre objetos	3
Relaciones entre clases	2
Introducción al modelado de software	2
Excepciones	1
Patrones básicos de diseño	3
Introducción al diseño de software	1

7 Bibliografía

Obligatoria

Ambler, S. (2004). *The Object Primer. Agile Model-Driven Development with UML 2.0.* (3rd edition ed.). Cambridge: Cambridge University Press.

Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code.* Addison-Wesley.

Gosling, J., Joy, B., & Steele, G. *The Java Language Specification.*

Lethbridge, Timothy, & Laganière, R. (2001). *Object-Oriented Software Engineering.* Maidenhead: McGraw Hill.

Liang, Y. D. (2007). *Introduction to Java programming: comprehensive version* (6th edition ed.). Upper Saddle River: Pearson Prentice Hall.

Liskov, B. (2001). *Program Development in Java: Abstraction, Specification, and Object-Oriented Design.* Boston: Addison Wesley.

McConnell, S. (2004). *Code Complete* (2nd edition ed.). Redmond: Microsoft Press.

Meyer, B. (1997). *Object-Oriented Software Construction* (2nd edition ed.). Santa Barbara: Interactive Software Engineering.

Meyer, B. (2009). *Touch of Class. Learning to Program Well with Objects and Contracts.* Berlin: Springer Verlag.

Adicional

Eckel, B. (2006). *Thinking in Java* (4th ed.). Upper Saddle River, NJ: Prentice Hall.

Miles, R., & Hamilton, K. (2006). *Learning UML 2.0*. Sebastopol, CA: O'Reilly.
Weisfeld, M. A. (2013). *The object-oriented thought process* (4th ed.). Upper Saddle River, NJ: Addison-Wesley.

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley.

Bruegge, B., & Dutoit, A. H. (2010). *Object-oriented software engineering: using UML, patterns, and Java* (3rd ed.). Boston: Prentice Hall.

8 Profesor

Mauricio Avilés Cisneros tiene estudios de Ingeniería en Computación (ITCR) y Maestría en Educación con énfasis en Docencia (UAM). Ha laborado para la institución desde 2003, primero como analista de sistemas del Departamento de Administración de Tecnologías de Información y Comunicación (DATIC). Profesor de los cursos Computación para Administración y Sistemas de Información para Administración de la escuela de Administración de Empresas durante el periodo 2007-2011 y 2018 en adelante. Labora como profesor e investigador a tiempo completo en la Escuela de Computación desde 2012, donde ha impartido los cursos de Introducción a la Programación, Taller de Programación, Estructuras de Datos, Programación Orientada a Objetos, Elementos de Computación, Análisis y Diseño de Algoritmos, y Simulación de Sistemas Naturales.

Ubicación: Centro Académico de San José.
Teléfono oficina: 25509586
Correo electrónico: maviles@tec.ac.cr
Medio oficial electrónico: TEC-Digital (http://tecdigital.tec.ac.cr)
Consulta: Se realizará con cita previa, con horario a convenir según el profesor y el estudiante.
Horario de consulta:

Lunes	10:00-12:00 / 13:00-17:00
Miércoles	10:00-12:00
Viernes	10:00-12:00

Adicionalmente se brinda consulta mediante Discord y correo electrónico, en modalidad 24/7, limitado por las posibilidades del profesor.

9 Asistente

Daniela Aguilar Ramírez, estudiante de Ingeniería en Computación en el Campus Tecnológico Local San José. Disponible en Discord para cualquier consulta que tengan.
Teléfono: 8405-9784
Correo: daguilar22@estudiantec.cr